BAB II

LANDASAN TEORI

2.1 Kecerdasan Buatan (*Artifical Intellegence*)

2.1.1 Pengertian Kecerdasan Buatan

Kecerdasan buatan (*Artifical Intellegence*) merupakan inovasi baru didalam bidang ilmu pengetahuan. Kecerdasan buatan merupakan bagian dari ilmu pengetahuan komputer yang khusus ditujukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Bagian utama dari kecerdasan buatan adalah basis pengetahuan (*knowledge base*), yaitu suatu pengertian atau pemahaman tentang wilayah subjek yang diperoleh melalui pembelajaran dan pengalaman (Kristanto, 2003). Kecerdasan buatan memiliki kemampuan untuk menyimpan informasi dalam jumlah besar dan memprosesnya dengan kecepatan yang sangat tinggi yang dapat menandingi kemampuan manuasia.

Beberapa definisi dari kecerdasan buatan yaitu (Kristanto, 2004)

1. Kecerdasan buatan adalah cabang ilmu komputer yang berhubungan dengan studi dan kreasi sistem komputer yang mempertunjukkan beberapa bentuk kecerdasan.
2. Sistem yang mempelajari konsep-konsep baru dan tugas-tugas.
3. Sistem yang dapat berfikir dan menarik kesimpulan yang berguna bagi dunia di sekitar kita.
4. Sistem yang dapat mengerti bahasa dan memahami pemandangan visuil (*visual*).
5. Sistem yang melakukan tipe-tipe yang lain seperti presentasi yang membutuhkan kecerdasan manusia.

Kecerdasan buatan mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia. Aktivitas manusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah, pemahaman bahasa alami dan sebagainya. Teknologi kecerdasan buatan dapat dipelajari dalam berbagai bidang bidang seperti robotika (*robotics*), penglihatan komputer (*komputer vision*), pengolahan bahasa alami (*natural language processing*), pengenalan pola (*pattern recognitiion*), system syaraf buatan (*artificial neural system*), pengenalan suara (*speech recognition*) dan system pakar (*expert system*). (Simarmata, 2006).

2.1.2 Ruang Lingkup Utama Kecerdasan Buatan

Ruang lingkup utama dalam kecerdasan buatan diantaranya adalah: sistem pakar, pengelolaan bahasa alami, pengenalan ucapan, robotika dan sensor, komputer vision, *Intelegent Computer-aided Intruction* dan game playing (kesuma dewi, 2003).

Sistem Pakar (*Expert System*) digunakan sebagai sarana untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan permasalahan dengan meniru keahlian yang dimiliki oleh pakar. Dengan teknologi yang memanfaatkan kecerdasan buatan ini diharapkan *user* dapat berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari yaitu memanfaatkan pengolahan bahasa alami (*Natural Language Processing*). Selain itu juga ada pengenalan ucapan (*Speech Recognition*), melalui pengenalan ucapan diharapkan manusia dapat berkomunikasi dengan komputer menggunakan suara.

Lingkup utama kecerdasan buatan lainnya adalah robotika dan sistem sensor (*Robotics and Sensory System*) serta komputer vision. Dimana komputer vision merupakan kemampuan untuk menginterpretasikan gambar atau objek-objek tampak melalui komputer. *Intelligent Computer-aided Instruction* yaitu komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar. Dan terakhir adalah *game playing*, dimana pembuatan game sudah memanfaatkan sistem kerja kecerdasan buatan.

2.1.3 Konsep Dasar Kecerdasan Buatan

Ada beberapa konsep dasar yang harus dipahami dalam kecerdasan buatan(*artifical intellegence*), diantaranya adalah (Kusrini, 2006):

1. *Turing Test* Metode Pengujian Kecerdasan

   Turing Test merupakan sebuah metode pengujian kecerdasan yang dibuat oleh Alan Turing. Proses uji ini melibatakan seorang penanya (manusia) dan dua obyek yang ditanyai. Yang satu adalah seorang manusia dan yang satunya adalah sebuah mesin yang akan diuji. Penanya tidak dapat melihat langsung kepada obyek yang ditanyai. Penanya diminta untuk membedakan mana jawaban komputer dan mana jawaban manusia berdasarkan jawaban kedua obyek tersebut. Jika penanya tidak dapat membedakan mana jawaban mesin dan mana jawaban manusia maka Turing berpendapat bahwa mesin yang diuji tersebut dapat diasumsikan CERDAS.

2. Pemrosesan Simbolik

   Komputer semula didesain untuk memproses bilangan atau angka-angka (pemrosesan numerik). Sementara manusia dalam berfikir dan menyelesaikan masalah lebih bersifat simbolik, tidak didasarkan pada sejumlah rumus atau melakukan komputasi matematika. Sifat penting dari AI adalah bahwa AI merupakan bagian dari ilmu komputer yang melakukan proses secara simbolik dan non-algoritmik dalam penyelesaian masalah.

3. *Heuristic*

   Istilah *heuristic* diambil dari bahasa Yunani yang berarti menemukan. *Heuristic* merupakan suatu strategi untuk melakukan proses pencarian (*search*) ruang problem secara selektif, yang memandu proses pencarian yang kita lakukan disepanjang jalur yang memiliki kemungkinan sukses paling besar.

4. Penarikan Kesimpulan (*Inferencing)*

   AI mencoba membuat mesin memiliki kemampuan berfikir atau mempertimbangkakn (*reasoning*). Kemampuan berfikir (*reasoning*) termasuk didalamnya proses penarikan kesimpulan (*inferencing*)

berdasarkan fakta-fakta dan aturan dengan menggunakan metode *heuristic* atau pencarian lainnya.

5. Pencocokan Pola (*Pattern Matching*)

   AI bekerja dengan metode pencocokan pola (*pattern matching*) yang berusaha untuk menjelaskan obyek, kejadian (*event*) atau proses, dalam hubungan logika atau komputasional.

2.2 Representasi Pengetahuan

Representasi pengetahuan adalah suatu teknik atau metode untuk merepresentasikan basis pengetahuan yang diperoleh ke dalam suatu skema atau diagram tertentu sehingga dapat diketahui relasi atau keterhubungan antara suatu data dengan data yang lain sehingga dapat diuji kebenaran penalarannya. Representasi pengetahuan dimaksudkan untuk mengorganisasikan pengetahuan dalam bentuk tertentu (Simarmata, 2006).

Adapun karakteristik dari metode representasi pengetahuan adalah :

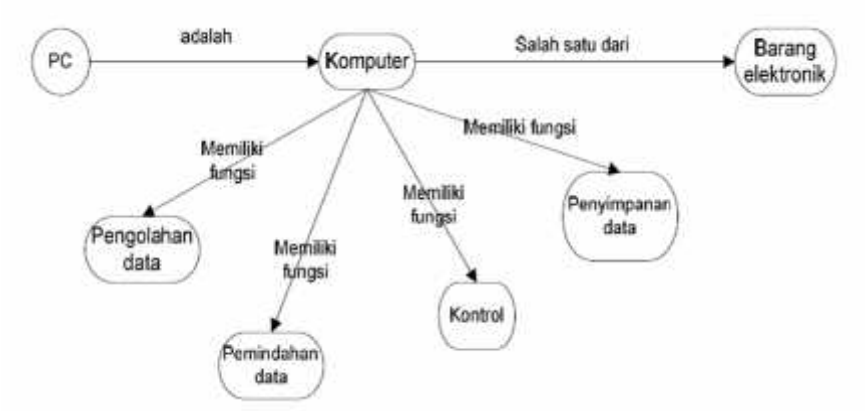
1. Harus bisa diprogram dengan bahasa pemrograman dan hasilnya disimpan dalam memori.
2. Dirancang sedemikian rupa sehingga isinya dapat digunakan untuk proses penalaran.
3. Model representasi pengetahuan merupakan sebuah struktur data yang dapat dimanipulasi oleh mesin inferensi dan pencarian untuk aktivitas pencocokan pola.

2.2.1 Semantik

Representasi pengetahuan semantik dibangun oleh M.R.Quillian, sebagai model memori manusia. Jaringan semantik adalah teknik representasi pengetahuan yang digunakan untuk informasi proporsional. Informasi proporsional adalah pernyataan yang mempunyai nilai benar atau salah. Misalnya, sebuah bujur sangkar mempunyai empat sisi. Informasi proporsional merupakan bahasa deklaratif karena mempunyai fakta (Simarmata, 2006).

Representasi jaringan semantik merupakan penggambaran grafis dari pengetahuan yang memperlihatkan hubungan hirarkis dari objek-objek. Komponen dasar untuk merepresentasikan pengetahuan dalam bentuk jaringan

semantik adalah simpul (*node*) dan penghubung (*link*). Objek direpresentasikan oleh simpul. Hubungan antar objek-objek dinyatakan oleh penghubung yang diberi label untuk menyatakan hubungan yang direpresentasikan.



Gambar 2.1 Jaringan Semantik

2.3 Pencocokan Pola (*Pattern Matching*)

Pencocokan Pola atau *Pattern Matching* adalah suatu metode yang digunakan untuk mencocokkan suatu pola tertentu (kumpulan huruf) dengan suatu kumpulan kata (teks) atau *string*. Pada bidang sains komputer metode *pattern matching* sangat banyak digunakan antara lain *Editor Teks*, Mesin Pencari *Web*, Analisis Gambar dan lain-lain. *String* dapat kita asumsikan sebagai kumpulan dari beberapa karakter yang membentuk suatu kesatuan. (Budiasa, 2009).

2.3.1 *Case Base Reasoning*

Case Base Reasoning (CBR) adalah metode utama yang diterapkan pada mesin inferensi *chatbot learning system*. Secara garis besar, konsep dari *case based reasoning* adalah menyelesaikan suatu masalah berdasarkan pengalaman memecahkan masalah/kasus yang mirip di masa lalu(Sri Mulyana, 2009).

Case Base Reasoning dapat memiliki makna yang berbeda, tergantung tujuan dari penalaran berupa penyesuaian dan penggabungan solusi sebelumnya untuk menyelesaikan sebuah masalah baru, menjelaskan terhadap solusi berdasarkan kasus sebelumnya, menemukan alasan dari kondisi sebelumnya untuk memahami situasi baru atau membangun sebuah solusi yang disepakati

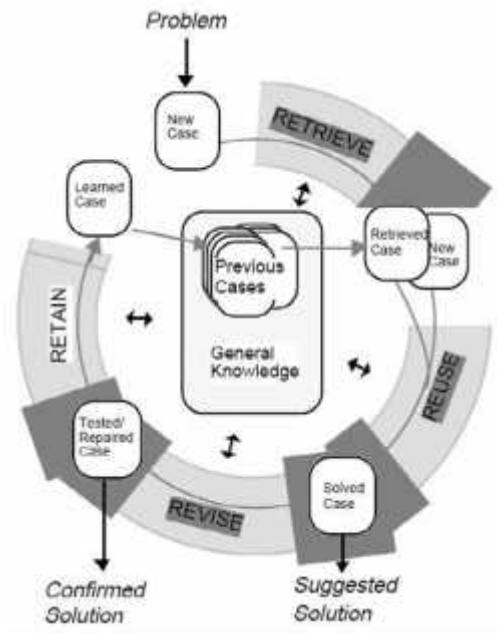
berdasarkan kasus sebelumnya. Dari beberapa aspek yang berbeda tersebut dapat dikelompokkan ke dalam dua tipe utama : *case base reasoning* interpretatif dan *case base reasoning* untuk penyelesaian masalah

Pada *case base reasoning* interpretatif, aspek yang penting adalah argumentasi apakah suatu situasi baru seharusnya diperlakukan seperti sebelumnya berdasarkan persamaan dan perbedaannya, sedangkan *case base reasoning* untuk penyelesaian masalah, bertujuan mendapatkan penyelesaian masalah baru dengan melakukan adaptasi terhadap penyelesaian pada kasus-kasus sebelumnya. Pembagian ini tidak dimaksudkan untuk memisahkan antar keduanya, karena pada kenyataannya banyak masalah yang memiliki kedua tipe tersebut, bahkan kebanyakan aplikasi pembelajaran berbasis kasus menggunakan kedua metode tersebut.

Secara singkat, tahap-tahap penyelesaian masalah berbasis *case base reasoning* adalah sebagai berikut : (Mantaras dkk,2006)

- Pengambilan kembali kasus-kasus yang sesuai dari memori (hal ini membutuhkan pemberian indeks terhadap kasus-kasus dengan menyesuaikan fitur-fiturnya).
- Pemilihan sekelompok kasus-kasus yang terbaik.
- Memilih atau menentukan penyelesaian.
- Evaluasi terhadap penyelesaian (hal ini dimaksudkan untuk meyakinkan agar tidak mengulang penyelesaian yang salah)
- Penyimpanan penyelesaian kasus terbaru dalam penyimpan kasus/memori.

Sesuai dengan tahap-tahap tersebut, Aamodt dan Plaza (Aamodt dan Plaza, 1994) menjelaskan sebuah *case base reasoning* sebagai sebuah siklus yang disingkat 4 R yaitu, *Retrieve, Reuse, Revise dan Retain* seperti pada gambar 2.2 berikut ini :



Gambar 2.2 Siklus *Case Base Reasoning* (Aamodt dan Plaza, 1994)

Penerapan *case based reasoning* pada chatbot di implementasikan pada alur *bot program* dan *brain file*, untuk penjelasannya sebagai berikut :

1. Siklus *retrieve* dilakukan pencarian kesamaan (*similarity*) antara kasus yang sedang ditangani (*new case*) berupa kosakata yang di *input*kan dalam bentuk pertanyaan dengan kosakata yang tersimpan didalam basis pengetahuan. Kosakata tersimpan yang paling mirip dengan *new case* akan dijadikan patokan sebagai *retrieved case.*
2. Siklus *reuse*, kosakata yang pernah diterapkan pada *retrieved case* dijadikan sebagai prediksi kosakata yang terpilih berupa solusi kosakata yang diusulkan (*suggested solution*) bagi kasus yang sedang ditangani. Kosakata yang terpilih tersebut dicobakan sebagai solusi untuk selanjutnya di konfirmasi untuk ketepatan calon jawaban dari pertanyaan yang di *input*kan untuk menjadi solusi (jawaban) yang tepat bagi *new case*.
3. Siklus *revise*, konfirmasi revisi tentang kosakata yang di usulkan akan dijadikan patokan untuk di pelajari berupa pencocokan pola dari kosakata yang terpilih dengan jawaban yang sesuai. Pada siklus ini akan dilakukan pembelajaran (*learning*) agar pada sesi berikutnya system *chatbot* dapat memberikan jawaban yang tepat bagi pertanyaan

yang sama. Jika konfirmasi jawaban yang diterima adalah tepat, maka calon jawaban tersebut akan diresmikan menjadi jawaban yang tepat bagi pertanyaan yang sedang ditangani (*new case*) setelah dipastikan mendapatkan solusi yang tepat,

4. Siklus *retain* selanjutnya pertanyaan tersebut disimpan untuk referensi bagi pertanyaan-pertanyaan yang mirip yang ditemukan pada sesi selanjutnya.

2.3.2 *Regular Exspression*

Regular Expression (*RE*) merupakan suatu konsep pencocokan pola (*pattern matching*) di dalam suatu *string*, dalam implementasinya *regular expression* menjadi suatu pola karakter yang banyak didukung oleh banyak aplikasi dan bahasa pemograman. Konsep *regular expression* yang akan mengenali pola dari karakter atau kata dan kemudian melakukan substitusi, *regular expression* biasa digunakan untuk menemukan sebagian kata yang sesuai dengan pola yang sudah ditentukan di dalam rangkaian kata yang ada. *Regular expression* juga bisa digunakan untuk menentukan apakah sebuah masukan yang diberikan sesuai dengan pola yang sudah ditentukan pada sebuah sistem.(Muhsin Shodiq,2011).

Penerapan *regular expression* pada *chatbot* dijelaskan seperti berikut, Salah satu trik yang dilakukan adalah dengan mencari kata atau frasa yang biasa digunakan dalam percakapan sehari-hari. Teknik yang sama juga digunakan untuk mengenali sebuah obyek.

2.3.3 Algoritma *Brute Force*

Algoritma *Brute Force* merupakan algoritma pencarian string termudah. Dengan asumsi bahwa teks berada di dalam *array T*[1..*n*] dan *pattern* berada di dalam *array P*[1..*m*], maka algoritma *brute force* dalam mencocokan *string* adalah sebagai berikut:

1. Mula-mula *pattern P* dicocokkan pada awal teks *T*.
2. Dengan bergerak dari kiri ke kanan, bandingkan setiap setiap karakter di dalam *pattern P* dengan karakter yang bersesuaian di dalam teks T sampai:
    a. Semua karakter yang dibandingkan cocok atau sama (pencarian berhasil), atau.

b. Dijumpai sebuah ketidakcocokan karakter (pencarian belum berhasil)

3. Bila *pattern P* belum ditemukan kecocokannya dan teks *T* belum habis, geser *pattern P* satu karakter ke kanan dan ulangi langkah 2.

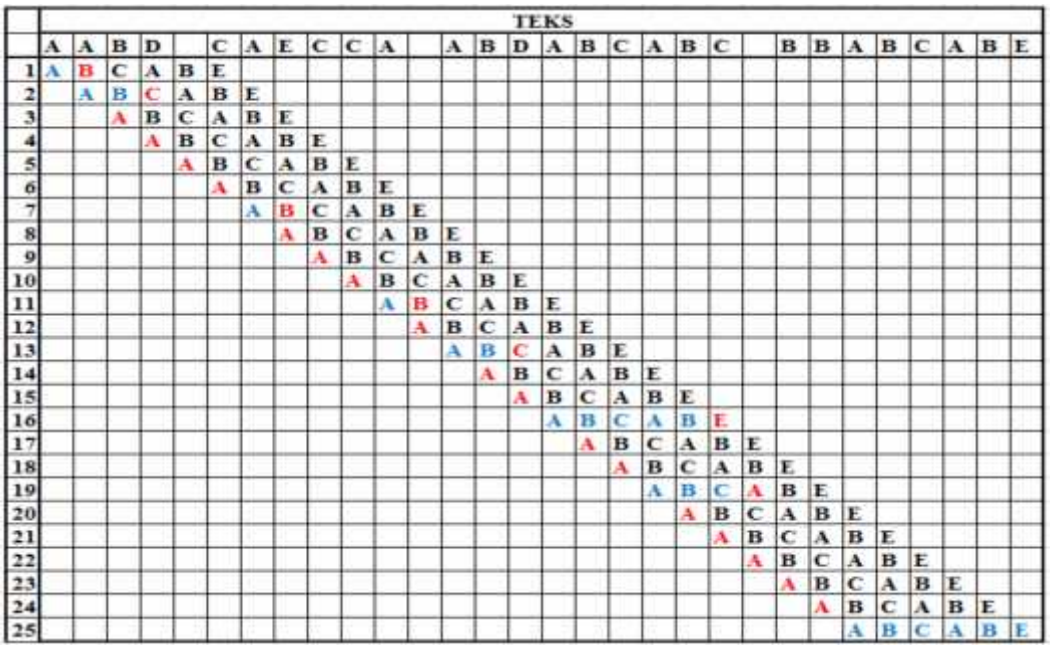
Maka dalam pencarian *string* dirumuskan sebagai berikut:

1. Teks (*text*), yaitu (*long*) *string* yang panjangnya *n* karakter
2. *Pattern*, yaitu *string* dengan panjang *m* karakter (*m* < *n*) yang akan dicari di dalam teks.

Langkah Pencocokan Pola algoritma *Brute Force* ini adalah sebagai berikut :

Teks : AABD CAECCA ABDABCABC BBABCABE

Pattern : ABCABE



Gambar 2.3 Langkah Pencocokan Pola Algoritma *Brute Force*

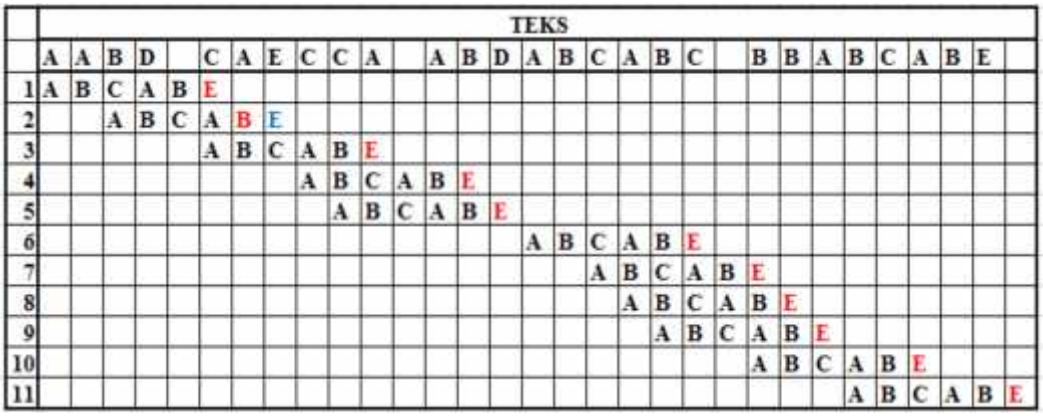
2.3.4 Algoritma *Boyer Moore*

Algoritma *Boyer Moore* dikembangkan oleh Bob Boyer dan J. Strother Moore pada tahun 1977. Pada berbagai literature dan referensi, algoritma ini menjadi standar *benchmark* dalam pencarian string. *Boyer Moore* melakukan hal yang sebaliknya, yaitu membandingkan karakter pattern dari kanan ke kiri.

Langkah Pencocokan Pola algoritma *Boyer Moore* ini adalah sebagai berikut :

Teks : AABD CAECCA ABDABCABC BBABCABE

Pattern : ABCABE



Gambar 2.4 Langkah Pencocokan Pola Algoritma *Boyer Moore*

2.3.5 Algoritma *Knuth Morris Pratt* (*KMP*)

Algoritma *Knuth Morris Pratt* (*KMP*) adalah salah satu algoritma pencarian string yang dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R. Pratt. Dengan menggunakan algoritma ini kita dapat mencari sebuah sub-string dari sebuah string dengan cara mencocokkan masing-masing karakter. Berbeda dengan algoritma *brute force* dimana jika pada saat pencocokkan *pattern* yang dicocokkan tidak sesuai maka bergeser satu karakter ke kanan, pada algoritma *KMP* sebelum pencocokkan dimulai algoritma ini menyimpan informasi yang dapat digunakan untuk mengetahui dimana pencocokkan berikutnya akan dilakukan, sehingga pergeseran yang tidak perlu tidak akan terjadi. Selain itu pada algoritma *KMP*, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter seperti halnya pada algoritma *brute force* (Rizki, 2008).

Menghitung fungsi pinggiran adalah langkah awal dan langkah terpenting dari algoritma *KMP*, karena fungsi pinggiran ini adalah kunci dari algoritma *KMP*. Hasil perhitungan fungsi pinggiran akan mengindikasikan pergeseran yang harus dilakukan *pattern*. Dengan adanya fungsi pinggiran ini pergeseran yang tidak perlu dapat dicegah. Fungsi pinggiran ini hanya bergantung pada karakter-karakter di dalam *pattern*, oleh karena itu kita dapat melakukan perhitungan sebelum proses pencarian string dimulai (Rizki, 2008). Untuk algoritma fungsi pinggiran dapat dilihat pada Gambar 2.4, dan untuk algoritma pencocokan pola dapat dilihat pada Gambar 2.5.

```
procedure HitungPinggiran(input m : integer, P : array[1..j] of char,
                          output b : array[1..j] of integer)
{ Menghitung nilai b[1..m] untuk pattern P[1..j] }
Deklarasi
  k,q : integer
Algoritma:
  b[1]←0
  q←2
  k←0
  for q←2 to m do
    while ((k > 0) and (P[q] ≠ P[k+1])) do
      k←b[k]
    endwhile
    if P[q]=P[k+1] then
      k←k+1
    endif
    b[q]=k
  endfor
```

Gambar 2.5 Algoritma fungsi pinggiran (*KMP*)

```
procedure KMPsearch(input m, n : integer, input P : array[1..m] of char,
                    input T : array[1..n] of char,
                    output idx : integer)
{ Mencari kecocokan pattern P di dalam teks T dengan algoritma Knuth-Morris-Pratt. Jika
ditemukan P di dalam T, lokasi awal kecocokan disimpan di dalam peubah idx. Masukan:
pattern P yang panjangnya m dan teks T yang panjangnya n.  Teks T direpresentasikan sebagai
string (array of character)
Keluaran: posisi awal kecocokan (idx). Jika P tidak ditemukan, idx = -1.}
Deklarasi
  i, j : integer
  ketemu : boolean
  b : array[1..m] of integer
  procedure HitungPinggiran(input m : integer, P : array[1..m] of char, output b :
 array[1..m] of integer)
  { Menghitung nilai b[1..m] untuk pattern P[1..m] }
Algoritma:
   HitungPinggiran(m, P, b)
  j←0
  i←1
  ketemu←false
  while (i ≤ n and not ketemu) do
    while((j > 0) and (P[j+1]≠T[i])) do
      j←b[j]
    endwhile
    if P[j+1]=T[i] then
      j←j+1
    endif
    if j = m then
      ketemu←true
    else
      i←i+1
    endif
 endwhile
 if ketemu then
   idx←i-m+1  { catatan: jika indeks array dimulai dari 0, maka idx ê i-m }
 else
   idx←-1
 endif
```

Gambar 2.6 Algoritma pencocokan pola (*KMP*)

Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat pencocokan pola adalah sebagai berikut:

1. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
    1. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
    2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser pattern berdasarkan tabel, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Kompleksitas waktu dari algoritma *KMP* ditentukan oleh waktu yang dibutuhkan saat penghitungan fungsi pinggiran dan pada saat pencarian string. Untuk menghitung fungsi pinggiran dibutuhkan waktu O(m), sedangkan untuk pencarian string dibutuhkan waktu O(n), maka kompleksitas waktu dari algoritma *KMP* adalah O(m+n).

Dengan algoritma *KMP* ini, waktu pencarian dapat dikurangi secara signifikan. Algoritma *KMP* dikembangkan oleh D. E. Knuth, bersama-sama dengan J. H. Morris dan V. R. Pratt.

Cara kerja algoritma *KMP* ini adalah sebagai berikut (Wicaksono, 2009) :

1. Tentukan Teks dan Pattern, dimana Teks diinisialisasikan dengan T dan Pattern dengan P.

    T : AABD CAECCA ABDABCABC BBABCABE

    P : ABCABE

2. Pada Pattern, hitung fungsi pinggiran dimana fungsi pinggiran B(j) di definisikan sebagai ukuran awalan terpanjang dari Pattern P yang

merupakan akhiran dari P[1..j]. Algoritma pencarian fungsi pinggiran adalah sebagai berikut :

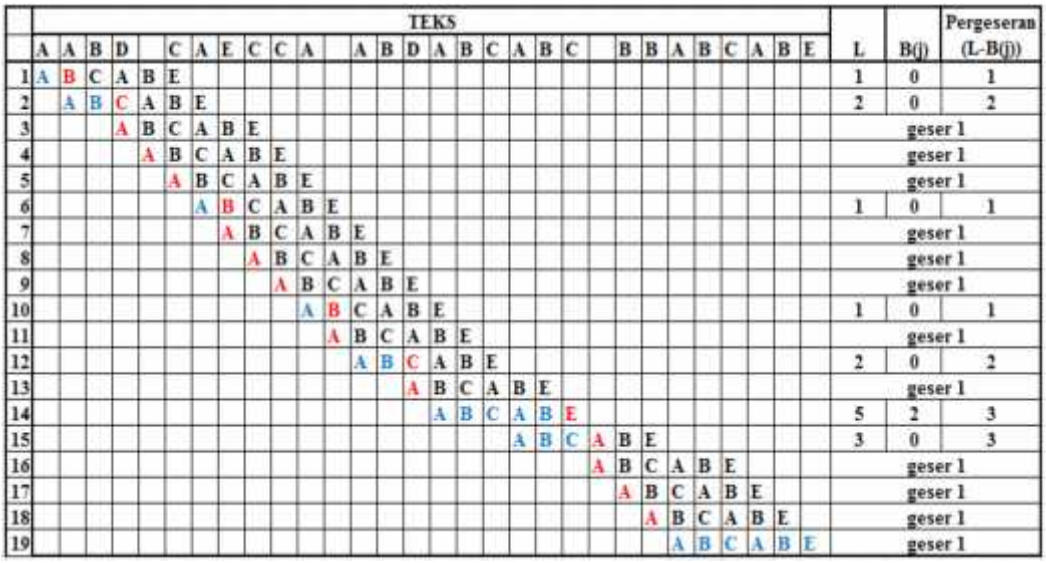
a. Inisialisasi j=1, q=2, k=0
b. Nilai B[1]=0
c. Bandingkan P[j] dengan P[q]

d. Jika k>0 dan p[q] tidak sama dengan p[k+1], maka k=b[k]
e. Jika sama, maka k=k+1
f. B[q]=k
g. q=q+1
h. Ulangi langkah ke 3-5 sebanyak panjang Pattern-1

Seperi berikut :

j	1	2	3	4	5	6
P(j)	A	B	C	A	B	E
B(j)	0	0	0	1	2	0

3. Lakukan pencocokan Pattern ke Teks dengan cara hitung L yaitu jumlah karakter Pattern yang sama dengan karakter Teks dikurangi nilai B(j). Lakukan pergeseran dari kiri ke kanan sebanyak nilai yang didapat.
4. Jika Pattern dan Teks tidak sama, maka lakukan pergeseran dari kiri ke kanan sebanyak satu karakter dapat dilihat pada Gambar 4.2 sebagai berikut :



Gambar 2.7 Langkah Pencocokan Pola Algoritma *KMP*

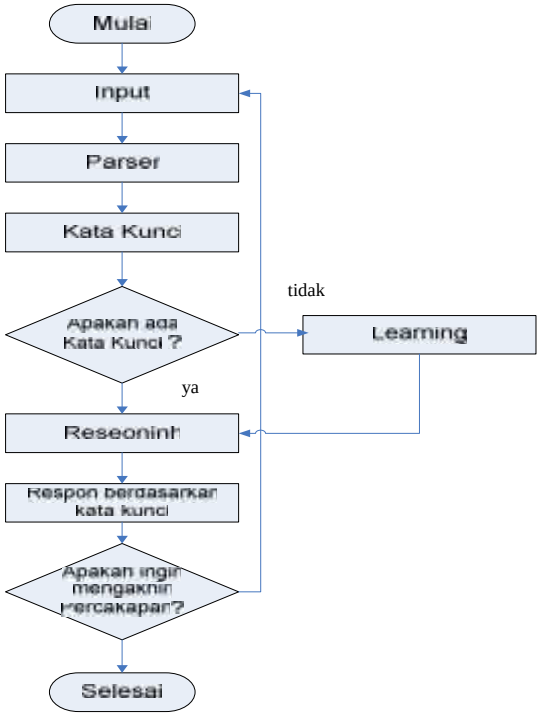
2.4 Chatbot

Chatbot adalah salah satu program kecerdasan buatan yang dirancang untuk dapat berkomunikasi langsung dengan manusia sebagai penggunanya, yang membedakan *chatbot* dengan sistem pemrosesan bahasa alami (*Natural Language Processing System*) adalah kesederhanaan algoritma yang di gunakan. Meskipun banyak *bots* yang tampaknya dapat menginterpretasikan dan menanggapi input manusia, sebenarnya *bots* tersebut hanya memindai kata kunci dalam input dan membalasnya dengan kata kunci yang paling cocok, atau pola kata-kata yang paling mirip dari basis data tekstual.

Chat dapat diartikan sebagai obrolan. *Bot* merupakan sebuah program yang mengandung sejumlah data dimana jika diberikan masukan maka akan memberikan jawaban. *Chatbot* dapat menjawab pertanyaan dengan membaca tulisan yang diketikkan oleh pengguna melalui *keyboard*. (Adriyani, 2004).

Pada mulanya, program komputer (*bots*) ini diuji melalui *turing test*, yaitu dengan merahasiakan identitasnya sebagai mesin sehingga dapat mengelabui orang yang berbicara dengannya. Jika pengguna tidak dapat mengidentifikasi *bots* sebagai suatu program komputer, maka *chatbot* tersebut dikategorikan sebagai kecerdasan buatan (*artificial intelligence*).

Dewasa ini, *chatbot* telah dimanfaatkan untuk tujuan praktis seperti bantuan online, layanan personal, atau akuisisi informasi, dalam hal ini dapat dilihat fungsi program sebagai suatu jenis agen percakapan (*conversational agent*).



Gambar 2.8 Alur *Chatbot.*

2.4.1 Perkembangan *Chatbot*

Sistem percakapan otomatis kini telah berkembang, dan perusahaan-perusahaan sudah menggunakan sitem-sistem tersebut untuk membantu call center memberikan panduan kontak. *Chatbot* pun sudah dimplementasikan melalui jejaring sosial, seperti *twitter* dan *Windows Live Messenger*. Portal *online* populer seperti *eBay* dan *PayPal* juga menggunakan agen *virtual multi* bahasa untuk memudahkan penggunanya. Misalnya, *PayPal* menggunakan *chatterbot* Louise untuk menangani *query* dalam bahasa Inggris dan *chatbot* Lea untuk *query* dalam bahasa Perancis. *Chatbot* tersebut Dikembangkan oleh VirtuOz, kedua agen tersebut menangani 400.000 percakapan setiap bulan setelah difungsikan pada September 2008 di situs *PayPal* (www.wikipedia.org). Selain itu *Chatbot* juga di implementasikan untuk bidang komersial, pendidikan, entertainment dan sektor pelayanan publik. (Kerly, A. Hall, P. & Bull, S. 2006.)

2.4.2 Komponen Utama *Chatbot*

Chatbot terdiri dari dua komponen utama yakni *bot program* dan *brain file*. *Bot program* merupakan program utama pada *chatbot* yang akan mengakses input dari pengguna, melakukan *parsing* dan kemudian membawanya ke *brain file (knowledge base)* untuk kemudian diberikan respon. Adapun *bot program* sendiri terdiri dari komponen *scanner* dan *parser*.

Brain file merupakan otak dari *chatbot* itu sendiri yang menentukan bagaimana cara *chatbot* berpikir dan akan memberikan respon. *Brain file* berfungsi sebagaimana tabel informasi (*knowledge base*) pada kompilator bahasa pemrograman tingkat tinggi. Di dalam *brain file* inilah disimpan semua kosakata, kepribadian, dan pengetahuan (*knowledge*) dari *chatbot*. Semakin banyak pengetahuan yang dimiliki *chatbot* maka akan semakin besar ukuran file dari *brain file* tersebut.

2.4.2.1 *Scanner*

Scanner merupakan salah satu bagian dari kompilator bahasa pada komputer yang bertugas melakukan analisis leksikal. *Scanner* menerima input berupa *stream* karakter kemudian memilah program sumber menjadi satuan leksik yang disebut dengan *token*. *Token* ini akan menjadi input bagi *parser*. Di dalam aplikasi *chatbot*, yang dimaksud dengan program sumber yang diolah oleh *scanner* adalah berupa kalimat input dari pengguna.

2.4.2.2 *Parser*

Parser atau *syntactic analyzer* pada kompilator bahasa pemrograman berfungsi untuk memeriksa kebenaran kemunculan setiap *token*. Pada *Chatbot system*, fungsi dari *parser* ini agak berbeda karena *token* yang akan diolah semua memiliki tipe yang sama yaitu berupa kata (*word*). Urutan kemunculan *token* yang berupa kata-kata tersebut akan diolah dengan mengacu pada *brain file* agar didapatkan makna kalimat yang sesungguhnya. Dengan kata lain, tahap analisa semantik terjadi di bagian *brain file*. Kemampuan dari *parser* untuk mengolah *token* dan bekerja sama dengan *brain file* inilah yang paling menentukan tingkat kecerdasan dari sebuah *chatbot*.

2.4.2.3 *Reasoning*

Reasoning adalah teknik penyelesaian masalah dengan cara mempresentasikan masalah ke dalam basis pengetahuan (*knowledge base*) menggunakan *logic* atau bahasa formal. Pada penerapan *chatbot* proses ini akan dikerahkan bila kata kunci terdapat dalam *knowledge base chatbot* dan dilakukan untuk mengembalikan respon ke pengguna. Proses ini menunjukkan bahwa masukan yang diberikan oleh pengguna tidaklah diproses sebagai satuan kata, tetapi sebagai kalimat utuh.

2.4.2.4 *Learning*

Learning merupakan pendefinisian aturan tertentu secara otomatis dalam menemukan aturan yang diharapkan bisa berlaku umum untuk data-data yang belum pernah kita ketahui. Pada penerapan *chatbot* proses *learning* dijalankan bila kata kunci pada masukan pengguna tidak terdapat dalam *knowledge base*. Kata kunci yang tidak ditemukan tersebut akan disimpan sebagai *dialog repository* untuk kemudian akan ditanyakan pada *bot program*.

2.4.3 Prinsip Kerja *Chatbot*

Bot program atau bagian aplikasi menentukan kemampuan dan keterampilan *chatbot* untuk berbicara pada anda atau pada pengguna lainnya, atau dengan kata lain *bot program* berperan sebagai mulut. Jika anda menjalankan *chatbot* untuk pertama kalinya maka ia akan seperti bayi yang baru lahir. *Chatbot* tidak tahu apa-apa, tidak punya nama dan tidak punya kepribadian. Untuk itu pengguna harus mengajarinya berbagai hal sehingga ia dapat berbicara dengan baik. Semua pelajaran tersebut dimasukkan ke *brain file*.

Metode *chatbot* untuk memberikan respon sebenarnya cukup sederhana seperti pada Eliza adalah *bot program* akan mencari pola kata tertentu pada *input* dari pengguna, dan memberikan respon sesuai dengan *output* yang telah dirancang sebelumnya. Oleh karena itu kita harus mengetahui apa saja yang ingin dibicarakan oleh pengguna dan memberikan respon yang sesuai di dalam *brain file*.

Sebagai contoh, pengguna mengeluh bahwa *chatbot* yang kita rancang berbicara omong kosong dengan mengetik kalimat: "*You are talking nonsense*". Agar *chatbot* dapat memberikan respon terhadap kalimat tersebut, kita dapat tambahkan baris berikut pada *brain file* dari *chat bot*:

- *You are talking nonsense* ← *trigger line*
  I am smarter than you ← *respon bot*

Dengan demikian jika pengguna sekarang *chatting* menggunakan kalimat yang disebutkan di atas maka *bot program* akan masuk ke dalam *brain file* dan kemudian memberikan respon dengan output yang sesuai dengan *trigger line: you are talking nonsense.*

Respon yang sama juga dapat muncul jika anda juga menuliskan baris berikut pada *brain file*:

- *Nonsense* ← *trigger line*
  I am smarter than you ← *respon bot*
  Respon yang sama juga akan muncul jika anda menuliskan:
- *Talking nonsense* ← *trigger line*
  I am smarter than you ← *respon bot*

- *Pinhead* ← *trigger line*
  I am smarter than you ← *respon bot*

Namun akan lebih menarik jika *chat bot* diberikan kemampuan untuk merespon kalimat yang beragam misalnya:

- *You are talking nonsense* ← *trigger line*
  I am smarter than you ← *respon bot*
- *Talking Nonsense* ← *trigger line*
  Talking is funny, isnt it? ← *respon bot*
- *Nonsense* ← *trigger line*
  Don't be stupid! ← *respon bot*
- *Pinhead* ← *trigger line*
  Please try not to be insulting,buddy ← *respon bot*

Dengan *brain file* seperti itu tentu *chatbot* akan terasa lebih hidup karena dapat memberikan respon dengan berbagai kalimat yang berbeda.

2.4.4 *Turing Test*

Seorang ahli matematika dari Inggris yang bernama Tahun 1950, Dr. Alan Turing dikenal sebagai "*Father of AI*" mengusulkan sebuah pengujian yang disebut *turing test* untuk kecerdasan buatan. Inti dari *turing test* ini adalah untuk menguji apakah sebuah mesin dapat meyakinkan atau menipu manusia sebagai lawan bicaranya bahwa mesin tersebut adalah seorang manusia, yang dimaksud mesin di sini adalah sebuah *natural language system* yang mampu meniru perilaku manusia, dalam hal ini mesin tersebut adalah berupa *chatbot* yang mampu meniru cara bicara manusia.

Pada *Turing test*, seorang manusia (sebagai juri) ditempatkan pada sebuah kamar, dan mesin atau manusia lainnya ditempatkan pada kamar yang lain. Juri dapat mengajukan pertanyaan atau menjawab pertanyaan dari mesin atau manusia di ruangan lainnya. Semua komunikasi dilakukan dengan melalui sebuah terminal, input dilakukan dengan cara mengetik kalimat-kalimat percakapan. Juri sebenarnya tidak mengetahui apakah lawan bicaranya tersebut adalah sebuah mesin atau manusia. Jika selama percakapan terjadi juri tidak bisa membedakan apakah yang dia ajak bicara adalah mesin atau manusia, maka mesin tersebut dinyatakan lulus dari *Turing test*.

2.5 Proses Belajar Mengajar

Proses belajar mengajar merupakan dua proses yang tidak dapat dipisahkan, belajar mengajar adalah interaksi atau hubungan timbal balik antara siswa dengan guru dalam proses pembelajaran. Inti dari proses pendidikan secara formal adalah mengajar sedangkan proses pengajaran adalah siswa belajar. Proses belajar tertumpu pada bagaimana guru memberi kemungkinan bagi siswa agar terjadi proses belajar yang efektif atau dapat mencapai hasil yang sesuai dengan tujuan (Subiyat Tartono, 2006).

2.5.1 Komponen Belajar Mengajar

Suatu sistem kegiatan belajar mengajar memiliki sejumlah komponen, yaitu (Subiyat Tartono, 2006) :

a. Bahan Pelajaran.

   Bahan pelajaran adalah substansi yang akan disampaikan dalam proses belajar mengajar.

b. Kegiatan Belajar Mengajar

   Segala sesuatu yang telah diprogramkan akan dilaksanakan dalam proses belajar mengajar.

c. Metode

   Metode adalah suatu cara yang dipergunakan untuk mencapai tuuan yang telah diciptakan.

d. Alat (media)

   Alat (media) adalah segala sesuatu yang dapat di gunakan dalam rangka mencapai tujuan pengajaran. Dalam hal ini *chatbot* dapat dijadikan sebagai alat bantu untuk mendukung penyampaian informasi pembelajaran.

e. Sumber Pembelajaran.

   Sumber pembelajaran adalah bahan / materi untuk menambah ilmu pengetehuan yang mengandung hal-hal baru bagi sipelajaran.

2.5.2 Metode Tanya Jawab

Metode tanya jawab adalah penyampaian pelajaran dengan jalan mahasiswa mengajukan pertanyaan dan dosen menjawab atau bisa juga suatu metode di dalam pendidikan di mana dosen bertanya sedang mahasiswa menjawab bahan atau materi yang ingin di perolehnya. Metode ini mempunyai beberapa manfaat, yaitu:

1. Melatih mahasiswa untuk mengemukakan atau menanyakan sesuatu yang menurutnya tidak atau kurang jelas.
2. Untuk mengarahkan pemikiran mahasiswa ke suatu kesimpulan (generalisasi).

3. Membangkitkan perasaan ingin tahu dan ingin bisa pada diri mahasiswa.

2.6 Program Berbasis *Web*

Pemrograman berbasis *web*, faktor yang menentukan kinerja aplikasi adalah kecepatan akses *database* dan kecepatan akses jaringan dan internet.

Aplikasi berbas Untuk menjalankannya, dibutuhkan engine tertentu, dalam hal ini *web server*.

2.6.2 *Apache*

Apache merupakan *web server* yang paling sering digunakan sebagai *server* internet dibandingkan *web server* lainnya. *Web server* merupakan *server* internet yang mampu melayani koneksi transfer di dalam protokol *HTTP* (*Hypertext Transfer Protocol*) saat ini *web server* merupakan inti dari *server-server* internet selain *e-mail server, ftp server*, dan *news server*.

2.6.3 *PHP*

PHP merupakan singkatan dari *Hypertext Preprocessor*, *PHP* digunakan untuk membuat aplikasi *web*, *PHP* mendukung banyak *database* (*MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC*, dll.).

PHP adalah bahasa (*scripting language*) yang dirancang secara khusus untuk penggunaan pada *web*. *PHP* adalah *tool* untuk pembuatan halaman *web* dinamis. Kaya akan fitur yang membuat perancangan *web* dan pemograman lebih mudah.

Sintak bahasa *PHP* adalah sama seperti sintak C. Jadi jika Anda sudah berpengalaman dengan C maka Anda senang dengan *PHP*. *PHP* lebih sederhana dibanding C karena dia tidak menggunakan sebagian dari C yang sulit. *PHP* juga tidak memasukkan kemampuan pemograman *low-level* dari C karena *PHP* dirancang untuk program web sites dan tidak memerlukan kemampuan ini. Seperti bahasa pemograman web lainnya *PHP* memroses seluruh perintah yang berada dalam skrip *PHP* di dalam *web server* dan menampilkan outputnya kedalam *web browser klien*. *PHP* adalah bahasa *scripting* yang menghasilkan *output HTML* atau pun *output* lain sesuai keinginan pemrogram (misalnya : PDF dan lain-lain) yang dijalankan pada *server side*. Artinya, semua sintaks yang kita berikan akan

sepenuhnya dijalankan pada *server* sedangkan yang dikirimkan ke *browser* hanya hasilnya (*output*) saja.

2.6.4 *MySQL*

Bahasa *PHP* mempunyai kelebihan yaitu kompabilitasnya dengan berbagai macam jenis *database*, dukungan dengan berbagai macam jenis sistem operasi. *PHP* lebih cocok dan umum digunakan jika di gabungkan dengan *database MySQL*.

Bahasa SQL pada umumnya informasi tersimpan dalam tabel-tabel yang secara logika merupakan struktur dua dimensi terdiri dari baris (*row* atau *record*) dan kolom (*column* atau *field*). Sedangkan dalam sebuah *database* dapat terdiri dari beberapa *table*.

Mysql cepat dan mudah untuk digunakan (*easy-to-use*) dan sebagai sistem manajemen *database* relasional (RDBMS) yang digunakan untuk *database* pada beberapa web site. Kecepatan adalah fokus utama pada pengembangan awal *MySQL*. Demi kepentingan kecepatan ini, mereka membuat keputusan untuk menawarkan fitur lebih sedikit dibanding pesaing utama mereka (sebagai contoh, *Oracle* dan *Sybase*). *MySQL* adalah lebih mudah dalam instalasi dan penggunaanya dibanding pesaing komersialnya.

2.7 Penelitian sebelumnya untuk *Chatbot*.

Penerapan *Chatbot* sebagai alat bantu media pembelajaran memiliki beberapan referensi, salah satunya referensi dari *review* penelitian terdahulu terhadap tugas akhir mahasiswa UIN Suska Riau, dimana *review* tersebut berguna untuk memberikan masukan dan ide untuk pembuatan tugas akhir yang akan dibuat. Tugas akhir yang menjadi bahan penelitian terdahulu (Hendra, 2012) adalah "Rancang Bangun Sistem Pembelajaran *Chatbot* Menggunakan Metode *Case Base Reasoning (Studi Kasus: Mata Kuliah Pengantar Teknologi Informasi)*" yang disusun oleh Hendra K.R Lase. Dalam tugas akhir tersebut memiliki perbedaan dengan tugas akhir yang akan dibuat, dimana pada pebelitian sebelumnya untuk penerapan pencocokan pola atau *pattern matching* menggunakan *Regular Expression*. *Regular Expression* adalah merupakan konsep pencocokan pola (*pattern Matching*) yang menggunakan konsep yang sudah baku berupa *source code* yang sudah disediakan oleh bahasa pemograman. Dan pembuatan *Chatbot* tersebut berbasis *Web*. Sedangkan tugas akhir yang akan dibuat adalah penerapan algoritma pencocokan pola atau *pattern matching* Menggunakan *Metode Knuth Morris Pratt* (KMP) untuk pembuatan Sistem Pembelajaran *Chatbot* pada mata kuliah Sistem Informasi. Dimana pencocokan yang dilakukan adalah pencocokan kata kunci sebagai pattern dengan pertanyaan yang diinputkan oleh mahasiswa. Dan Sistem Pembelajaran Chatbot ini akan dibuat berbasis *Web* dengan menggunakan bahasa pemograman *PHP* dan *MySQL* sebagai *database* sistemnya.